MACHINE LEARNING

Subjective answer type questions.

1. R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?

Ans. R-squared is generally considered a better measure of goodness of fit in regression models compared to Residual Sum of Squares (RSS). R-squared indicates the proportion of variance in the dependent variable explained by the independent variables, providing a more comprehensive assessment of model fit, whereas RSS only measures the total deviation of observed values from predicted values, without considering how much of the variability in the dependent variable is explained by the model.

2. What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.

Ans. In regression analysis:

Total Sum of Squares (TSS) represents the total variation in the dependent variable around its mean.

Explained Sum of Squares (ESS) represents the variation in the dependent variable that is explained by the regression model.

Residual Sum of Squares (RSS) represents the unexplained variation in the dependent variable, which is the variation not accounted for by the regression model.

The equation relating these three metrics is given below:-

TSS = ESS + RSS

This equation illustrates that the total variation in the dependent variable (TSS) is decomposed into the variation explained by the regression model (ESS) and the residual unexplained variation (RSS).

3. What is the need of regularization in machine learning?

Ans. Regularization in machine learning is needed to prevent overfitting and improve the generalization ability of a model. Overfitting occurs when a model learns to capture noise and random fluctuations in the training data, leading to poor performance on unseen data. Regularization techniques introduce constraints on the model parameters during training, effectively penalizing overly complex models and encouraging simpler solutions. This helps to reduce overfitting and make the model more robust to variations in the data, resulting in better performance on unseen data. Regularization techniques such as L1 regularization (Lasso), L2 regularization (Ridge), and Elastic Net are commonly used in machine learning to achieve this goal.

4. What is Gini–impurity index?

Ans. The Gini impurity index is a measure used in decision tree algorithms to evaluate the impurity or randomness of a set of data points. It calculates the probability of incorrectly classifying a randomly chosen element in the dataset if it were randomly labeled according to the distribution of labels in the subset.

In simpler terms, it measures how often a randomly chosen element would be incorrectly classified if it were labeled randomly according to the distribution of labels in the subset. A lower Gini impurity indicates a purer node, meaning the data in that node is more homogeneous with respect to the target variable, which is desirable for decision tree algorithms.

5. Are unregularized decision-trees prone to overfitting? If yes, why?

Ans. Yes, unregularized decision trees are prone to overfitting. Decision trees have a tendency to create complex, deep trees that perfectly fit the training data, including noise and outliers. This can lead to poor generalization performance on unseen data because the model has essentially memorized the training data rather than learning the underlying patterns. Regularization techniques, such as pruning or limiting the maximum depth of the tree, are employed to mitigate overfitting by encouraging simpler tree structures.

6. What is an ensemble technique in machine learning?

Ans. An ensemble technique in machine learning involves combining multiple models to improve predictive performance. Instead of relying on a single model, ensemble methods leverage the diversity of multiple models to make more accurate predictions. Popular ensemble techniques include bagging, boosting, and stacking, each with its own approach to combining models effectively. The key idea behind ensemble techniques is to harness the collective wisdom of diverse models to achieve better overall performance than any individual model could achieve on its own.

7. What is the difference between Bagging and Boosting techniques?

Ans. The main difference between Bagging and Boosting techniques lies in how they combine multiple models:

Bagging : In Bagging, multiple models are trained independently on different subsets of the training data, typically sampled with replacement. The final prediction is made by averaging for regression or voting for classification the predictions of all individual models. Examples of bagging methods include Random Forest.

Boosting: In Boosting, models are trained sequentially, with each subsequent model learning from the mistakes of its predecessors. The focus is on improving the prediction of misclassified instances in the training set. Boosting assigns higher weights to misclassified instances, thereby allowing subsequent models to concentrate more on those instances. Examples of boosting methods include AdaBoost and Gradient Boosting Machines (GBM).

In summary, while both Bagging and Boosting techniques use multiple models to improve performance, Bagging trains models independently and combines their predictions, while Boosting trains models sequentially, with each subsequent model focusing more on correcting the errors made by previous models.

8. What is out-of-bag error in random forests?

Ans. The out-of-bag (OOB) error in random forests is an estimation of the model's performance on unseen data. In random forests, each decision tree is trained using a bootstrap sample of the original data, leaving out around one-third of the data (on average) which forms the out-of-bag samples. These out-of-bag samples are not used in the training of the corresponding tree. After training, each out-of-bag sample is passed through the trees for which it was left out, and the predictions are aggregated. The OOB error is then calculated as the average error across all out-of-bag samples. This provides a reliable estimate of the model's performance without the need for a separate validation set.

9. What is K-fold cross-validation?

Ans. K-fold cross-validation is a technique used in machine learning for assessing the performance of a predictive model. It involves splitting the dataset into K equal-sized folds, then training the model K times, each time using K-1 folds as the training data and the remaining fold as the validation data. This process allows for a more robust estimation of the model's performance by using different subsets of data for training and validation, helping to reduce the impact of variability in the data and providing a more reliable evaluation of the model's generalization ability.

10. What is hyper parameter tuning in machine learning and why it is done?

Ans. Hyperparameter tuning in machine learning involves the process of selecting the optimal hyperparameters for a given algorithm. Hyperparameters are parameters that are set prior to the training process and cannot be directly learned from the data, unlike model parameters. Examples of hyperparameters include the learning rate in gradient descent, the number of layers in a neural network, or the depth of a decision tree.

Hyperparameter tuning is crucial because the performance of a machine learning algorithm is highly dependent on the choice of hyperparameters. Selecting the right hyperparameters can significantly impact the model's performance, including its accuracy, generalization ability, and computational efficiency. Hyperparameter tuning is typically done through techniques like grid search, random search, or Bayesian optimization, aiming to find the optimal combination of hyperparameters that results in the best model performance on unseen data.

11. What issues can occur if we have a large learning rate in Gradient Descent?

Ans. If the learning rate in Gradient Descent is set too high, several issues can occur:

Divergence: The optimization process may fail to converge, leading to the model's parameters oscillating or diverging rather than converging to the optimal values.

Overshooting: The algorithm may overshoot the minimum of the loss function, causing it to oscillate around the minimum or even diverge.

Instability: Large learning rates can lead to unstable updates, causing the model's parameters to fluctuate widely with each iteration, making it challenging to find an optimal solution.

Poor Performance: High learning rates can prevent the model from effectively learning the underlying patterns in the data, resulting in poor performance on both the training and validation datasets.

To mitigate these issues, it's essential to carefully select an appropriate learning rate, often through techniques like learning rate schedules, adaptive learning rate methods, or cross-validation.

12. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

Ans. Logistic Regression is a linear classification algorithm, meaning it models the relationship between the independent variables and the log-odds of the dependent variable using a linear function. Therefore, it's inherently limited to linear decision boundaries.

For non-linear data, where the decision boundary is not linear, Logistic Regression may not perform well as it cannot capture the complex relationships between the features and the target variable. In such cases, more flexible models like Decision Trees, Random Forests, Support Vector Machines (with non-linear kernels), or neural networks are often preferred, as they can capture non-linear patterns in the data more effectively.

13. Differentiate between Adaboost and Gradient Boosting.

Ans. Adaboost and Gradient Boosting are both ensemble learning methods, but they differ in their approach to building the ensemble:

Adaboost (Adaptive Boosting): Adaboost focuses on sequentially training weak learners (e.g., decision trees) on repeatedly modified versions of the data. Each subsequent learner pays more attention to the instances that were misclassified by previous learners. The final prediction is a weighted sum of the predictions made by each weak learner.

Gradient Boosting: Gradient Boosting builds an ensemble of weak learners, typically decision trees, in a stage-wise manner. Unlike Adaboost, Gradient Boosting aims to minimize a loss function (e.g., mean squared error for regression, or log loss for classification) directly by adding weak learners that minimize the loss gradient. Each subsequent learner is trained to correct the errors of the previous ones.

In summary, while both Adaboost and Gradient Boosting leverage weak learners to build an ensemble, Adaboost focuses on modifying the data distribution to improve subsequent learners, while Gradient Boosting directly minimizes a loss function by adding weak learners that correct the errors made by previous ones.

14. What is bias-variance trade off in machine learning?

Ans. The bias-variance tradeoff in machine learning refers to the balance between two types of errors that affect a model's performance:

Bias: Bias refers to the error introduced by approximating a real-world problem with a simplified model. High bias models are overly simplistic and tend to underfit the data, meaning they fail to capture the underlying patterns in the training data.

Variance: Variance refers to the error introduced by the model's sensitivity to small fluctuations or noise in the training data. High variance models are overly complex and tend to overfit the data, meaning they capture noise in the training data as if it were true signal.

The tradeoff arises because reducing bias often leads to an increase in variance, and vice versa. The goal is to find the optimal balance between bias and variance to achieve the best generalization performance on unseen data. This balance is critical for building models that generalize well to new data while avoiding both underfitting and overfitting.

15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Ans. brief description of each kernel used in Support Vector Machines (SVM):

Linear Kernel: The linear kernel is the simplest kernel function. It computes the dot product of the input features, effectively creating a linear decision boundary. It works well when the data is linearly separable, meaning the classes can be separated by a straight line in the feature space.

RBF (Radial Basis Function) Kernel: The RBF kernel is a popular choice for SVMs as it can capture complex, non-linear decision boundaries. It maps the input features into a high-dimensional space and computes the similarity between data points based on their Euclidean distance. It is characterized by a single hyperparameter, the 'gamma' parameter, which controls the influence of each training example.

Polynomial Kernel: The polynomial kernel is used to handle non-linear decision boundaries by mapping the input features into a higher-dimensional space using polynomial functions. It computes the similarity between data points as the polynomial of the dot product between the input features. The degree of the polynomial is a hyperparameter that determines the flexibility of the decision boundary.